განმარტეთ map :: (a → b) → [a] → [b] ფუნქცია ორი გზით-სიის კონსტრუქტორით და რეკურსიულად.

map f xs = [f x | x ← xs] mymap f xs =foldr (\x acc->f x:acc) [] xs

map f [] = []

map f (x:xs) = f x : map f xs

filter :: (a → Bool) → [a] → [a]

filter p xs = [x | x ← xs, p x]

filter p [] = []

filter p (x:xs) myfilter p xs = foldr (\x acc->if p x then x:acc else acc) [] xs

| p x = x : filter p xs

| otherwise = filter p xs

all :: (a → Bool) → [a] → Bool

all p xs = and [p x | x ← xs]

any :: (a → Bool) → [a] → Bool

any p xs = or [p x | x ← xs]

foldr :: (a → b → b) → b → [a] → b

foldr f v [] = v

foldr f v (x:xs) = f x (foldr f v xs)

sum [7,8,9] = foldr (+) 0 [7,8,9]

რა შედეგს მოგვცემს

length = foldr (λ\_ n → 1+n) 0

reverse = foldr (λx xs → xs ++ [x]) []`

myconcat :: [[a]] -> [a]

myconcat xss = foldr (++) [] xss

myproduct = foldr1 (\*)

mylast :: [a] -> a

mylast = foldl1 (\\_ x -> x)

myhead :: [a] -> a

myhead = foldr1 (\x \_ -> x)

mmyfilter :: (a -> Bool) -> [a] -> [a]

mmyfilter p = foldr (\x acc -> if p x then x : acc else acc) []

myreverse :: [a] -> [a]

myreverse = foldl (\acc x -> x : acc) []

mymaximum :: (Ord a) => [a] -> a

mymaximum = foldr1 (\x acc -> if x > acc then x else acc)